Supplementary Note: Evaluating Variant Files

Evaluating Variant Files

This document considers the problem of assessing the performance of variant callers and shows the issues that arise in doing this correctly. It then goes on to show how `vcfeval` a novel tool to compare VCF files, correctly solves these issues. `vcfeval` is part of the freely available rtgTools package, available at: ftp://ftp-trace.ncbi.nih.gov/giab/ftp/tools/RTG/

1. Variant Evaluation.

When evaluating how correct a variant caller is, we assume that we have three pieces of information: a *reference* sequence (for example an assembled genome); a *baseline* set of variations on the reference; and a *called* set of variations on the reference. The called sequence will be the best possible one if it correctly includes everything in the baseline (the true positives), and has no incorrect calls (false positives) and no calls it has missed (false negatives).

The naïve way of computing these numbers is to look at the same reference locations in the baseline and called variant files and see if they are the same call at the same position. Unfortunately this is not sufficient; variant sequences can often be represented more than one way for indels and MNPs.

The simplest case is when there is an indel in the middle of a homopolymer repeat.

```
Reference       CAAAAAAG
```

Baseline variants = 1:CAA->C

Called variants = 5:AAA->A

```
Baseline        C..AAAAG
Called          CAAAA..G
```

If we “replay” these variants into the reference genome assembly used during calling in a uniform way, we have:

```
Baseline        CAAAAG
Called          CAAAAG
```

So despite a different representation the baseline and the called sequences are identical and will have a single true positive.

Consider a more complex example where there are dinucleotide repeats:

```
Reference = ACGTACCAGATATCACAACATATATATA
```

Baseline variants = 4:T->G, 9:GAT->G and 20:A->ATA

Called variants = 4:T->G, 11:TAT->T and 28:A->ATA

Looking just at the variant positions we will conclude there is 1 true positive, 2 false negatives (not in the called variant set) and 2 false positives (not in the baseline variant set). But if we replay the baseline and called variants we get two identical sequences:

```
Reference       ACGTACCAGATATCACAACATATATATA
Baseline        ACGGACCAG..ATCACAACATATATATATA
Called          ACGGACCAGAT..CACAACATATATATATA
```

After replay:

```
Baseline        ACGGACCAGATCACAACATATATATATA
Called          ACGGACCAGATCACAACATATATATATA
```

As seen above, after replaying the variants, we get identical sequences for the *baseline* and *called* variants. `vcfeval` will correctly report 3 true positives, 0 false positives and 0 false negatives.

A third type of problem occurs when MNPs are called different ways.

```
Reference      CAACGTAAG
```

Baseline variants = 4:C->T, and 7:A->C

Called variants = 4:ACGT->TGTC

```
Baseline        CAATGTCAG
Called          CAATGTCAG
```

Again the baseline and called variants when replayed give an identical sequence.

Of course all of these types of problems can occur mixed together. The situation can become even more complex when there are false positives or false negatives. There may be different ways of assigning the various calls to the different classes.

`vcfeval` aims to match the baseline and called sequences so as to maximize true positives and minimize false positives and false negatives. This is done in such a way that the number of true positives plus the number of false negatives equals the total number of calls in the baseline. Also for heterozygous calls to agree they must both be heterozygous and both alleles in the calls must agree.

There are three major steps in variant evaluation: i) Best path identification; ii) Linking variants; iii) Creation of ROC curve.

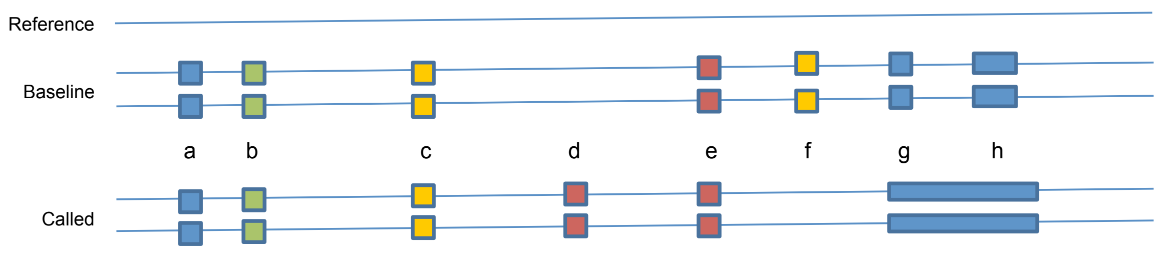
2. Path creation

A path through a call sequence is a selection of subset of calls. The idea is that the calls included as correct in the baseline and called paths will be equal (after being replayed). The calls excluded from the baseline will correspond to false negatives and the calls excluded from the called sequence will be classified as false positives. The calls in the baseline path and the called path (which always agree) are classified as true positives.

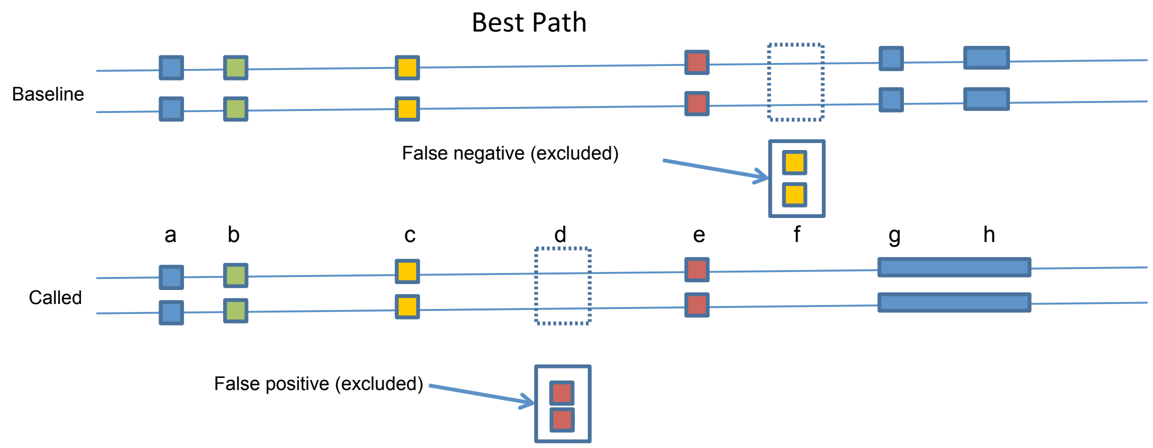
2.1 Best path identification

`Vcfeval` searches through all possible paths in both the baseline and called sequences and finds the pair of paths that maximize true positives and minimize errors. Potentially there are an exponential number of paths to be explores however in practice if the alternative (replayed) paths coincide at the same position on the reference, then the one which minimizes the number of errors up to that point can kept and the others discarded. In practice this happens frequently keeping the memory and processing requirements under control.

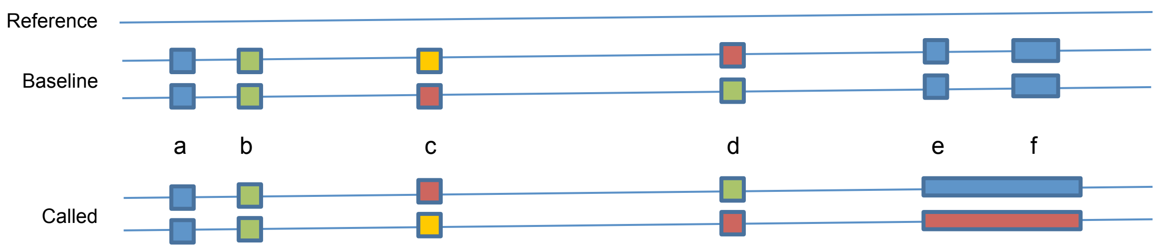
For example, given a set of simple diploid homozygous variants as shown below:



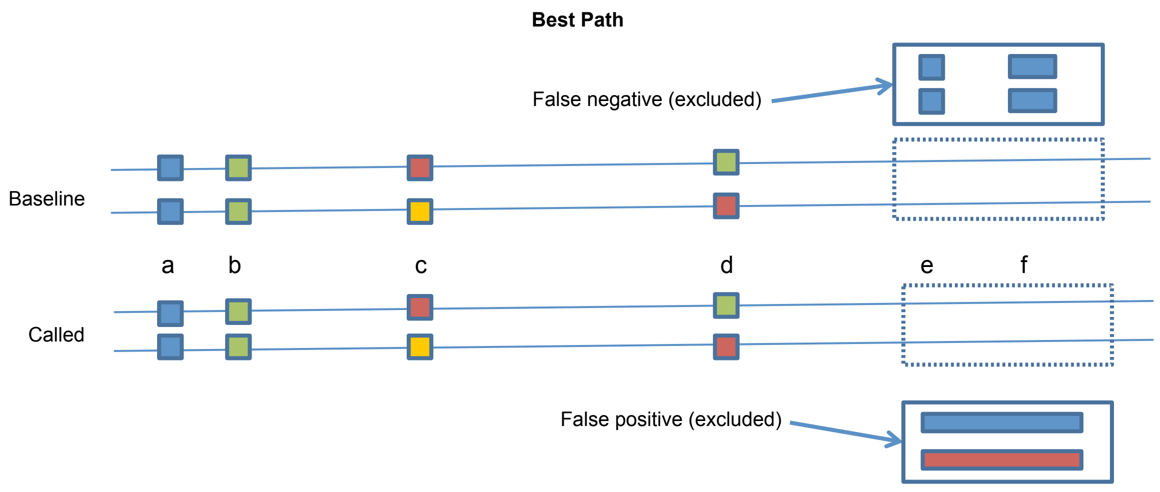
We then construct the best path by exploring all available paths through two sets of variants. For above example, the best path would include *a, b, c, e, g* and *h* and exclude *f* and *d*. Note that variants *g* and *h* in the baseline calls are combined as a single call in the called variant set. The following image displays the best paths for the case above:



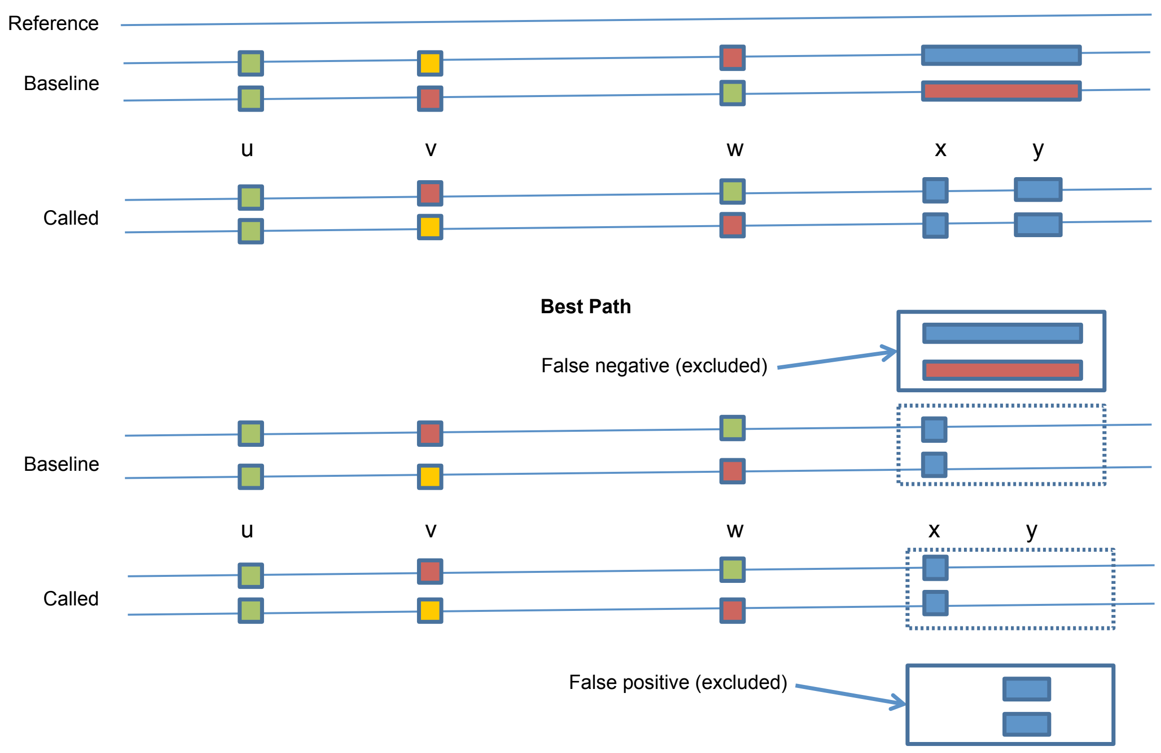
Similarly, we can construct a best path through heterozygous variants, currently all heterozygous variants are treated as non phased and evaluated accordingly, for example consider following sets of homozygous and heterozygous variants.



For the above case, we would consider variants *a, b, c* and d as true positive and *e* and *f* as false negative and *e* + *f* combined as a single false positive. Following is the best path for the above:



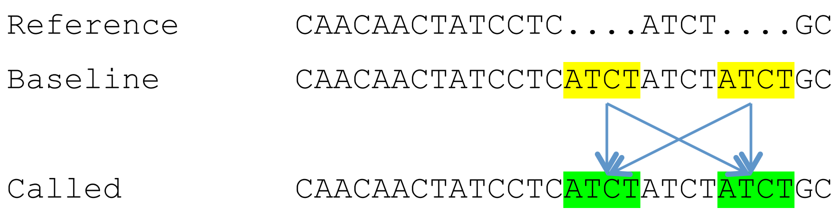
Note: Even if variant *e* is called correctly and *f* is called incorrectly in the combined call, the `vcfeval` will treat *e* and f as false negative and *e* + *f* combined as false positive. Conversely, in the case where we had 1 variant (*x* + *y*) in *baseline* variant set which was covered by 2 variants (*x* and *y*) in *called* variant set, if *x* was correct and *y* was incorrect in this case `vcfeval` will report *x* as true positive and *y* as false negative and false positive. The following figure shows this case:



3. Weighting

For generating a receiving operator characteristics (ROC) curve, the number of true positive and false positive variants must be counted. Generally each called variant will have a corresponding base line variant but due to the nature of complex calling there can be a 1 to many relationships between baseline and called variant. To keep number of true positives plus the number of false negatives equals to the total number of calls in the baseline, each called true positive call must be weighted.

In some cases, it is impossible to exactly how many called variant correspond to baseline variant, for example:



```
Reference      CAACAACTATCCTC....ATCT....GC
Baseline       CAACAACTATCCTCATCTATCTATCTGC

Called         CAACAACTATCCTCATCTATCTATCTGC
```

In the above case, due to the repeating nature of the reference the ATCT can equally correspond to first or the second occurrence in the baseline. To get around this, we *weight* variants between *sync points*. A *sync point* is a location where diploid baseline and called path are at the same position on the reference and they are not currently in the middle of any variant location.

For example:

```
Reference      ACAGTCACGG
Baseline       ACGGTCACTG
Called         ACGGTTACGG
```

The following *sync points* will be created for the above example (each vertical line represents a *sync point*):

```
Reference      AC | AGT | CAC | GG
Baseline       AC | GGT | CAC | TG
Called         AC | GGT | TAC | GG
```

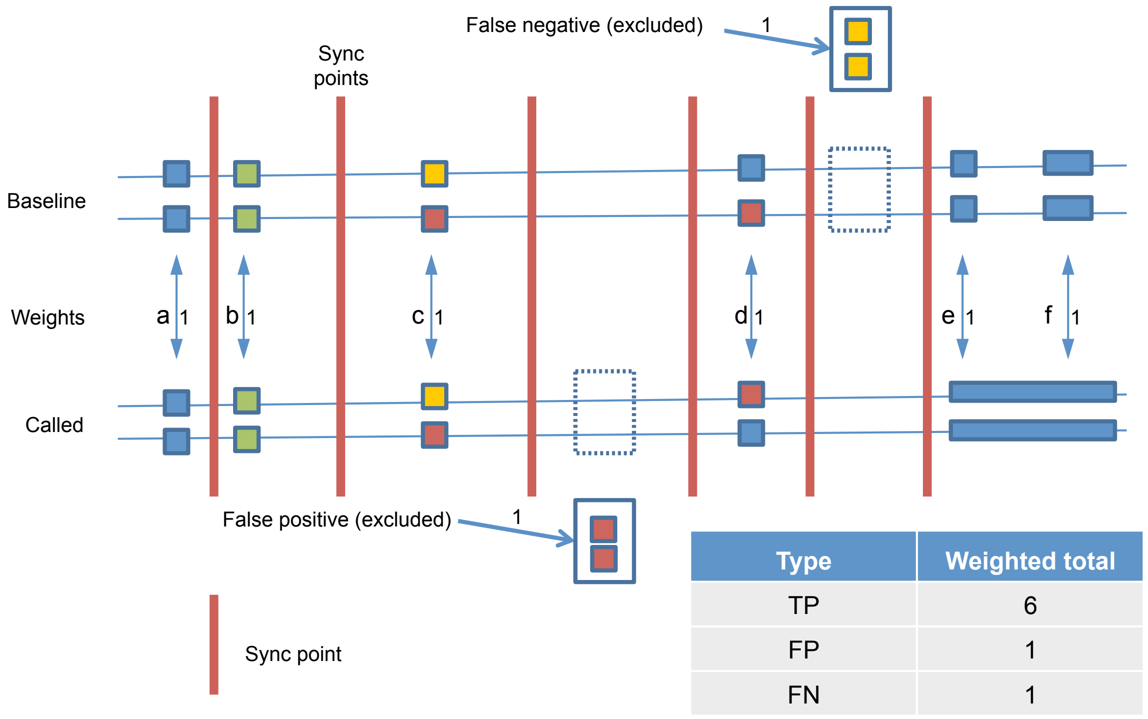
Note: an optimization in the best path creation skips all the locations which does not contain any variants thus the sync points occurs just before the next available variant.

Once all the sync points are created, each called variant is weighted using following formula:

$$w = \frac{B_{s_n - s_{n-1}}}{C_{s_n - s_{n-1}}}$$

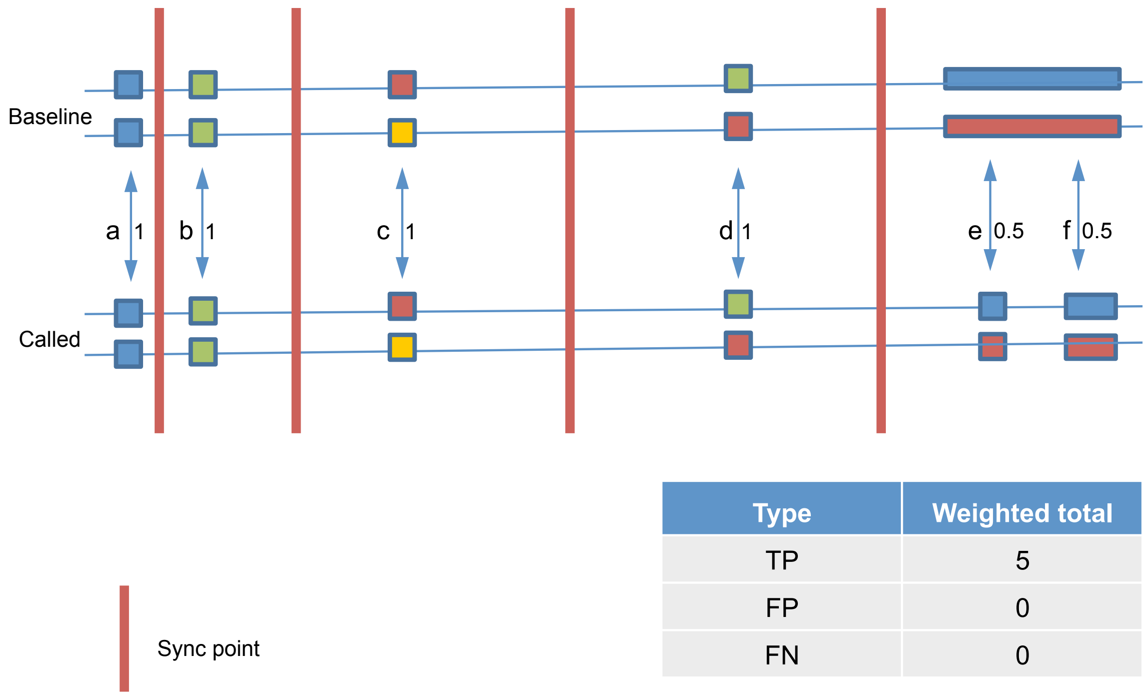
where B is the number of baseline variants between the current (S_n) and previous sync points (S_{n-1}) and C is the number of called variants between the current and previous sync points.

Following are some examples of weighting of true positives.



Type	Weighted total
TP	6
FP	1
FN	1

As seen above, all the *called* variants corresponds to exactly one *baseline* variants, note that the variant $e + f$ two variants but they are explained by only one *called* variant. In the case, where a *baseline* variant is explained by two *called* variants, then each call will have weight of 0.5. Following is the example of such case, baseline variant $e + f$ is explained by *called* variant *e* and *f*, and their weight is 0.5 each (1 baseline variant / 2 called variants).



Type	Weighted total
TP	5
FP	0
FN	0

4. Receiver operating characteristic (ROC) curve

In this step, the true positive and false positive sets are plotted over various threshold values. The threshold is usually the genotype quality (GQ) or AVR values from the VCF file. The x-axis lists the false positives in the called VCF, whereas the y-axis lists the true positives found in the called VCF, as compared to the baseline. Total counts or fractions can be used in the axes. ROC analysis provides a tool to select optimal call sets or parameters and to discard suboptimal ones. Often the area-under-the ROC curve can be used as metric of performance in these analyzes.

For more information, http://en.wikipedia.org/wiki/Receiver_operating_characteristic